DLCV hw2 report

系級：電機四　姓名：傅敬倫　學號:b06505011

Problem1 (no collaboration)

1.

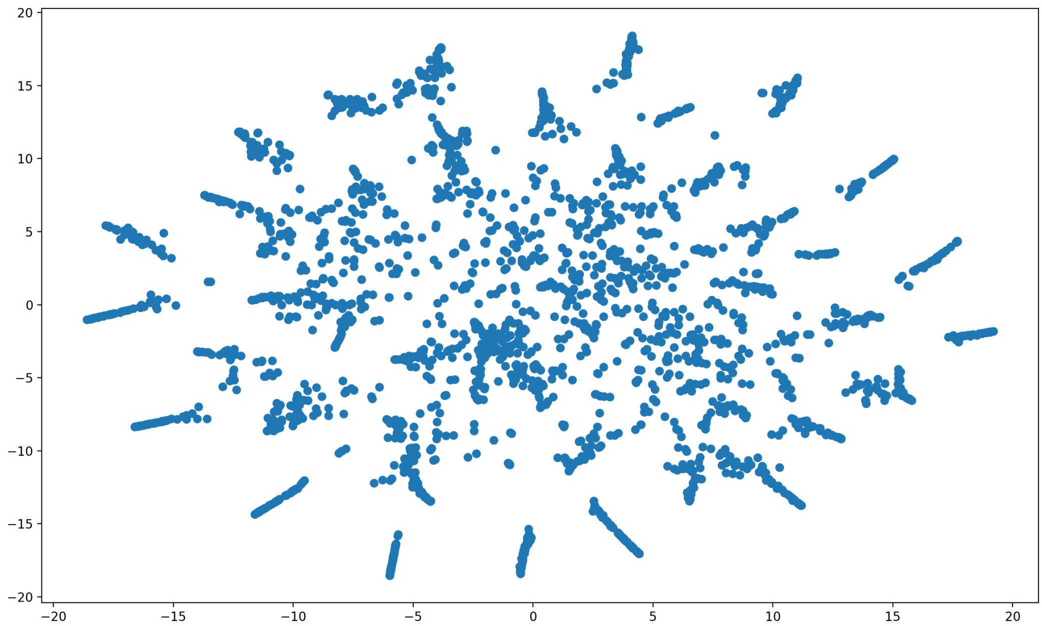
```
Net(
  (layer1): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace=True)
      (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): ReLU(inplace=True)
      (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (9): ReLU(inplace=True)
      (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (12): ReLU(inplace=True)
      (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (16): ReLU(inplace=True)
      (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (19): ReLU(inplace=True)
      (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (26): ReLU(inplace=True)
      (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (29): ReLU(inplace=True)
      (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (32): ReLU(inplace=True)
      (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (36): ReLU(inplace=True)
      (37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (39): ReLU(inplace=True)
      (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (42): ReLU(inplace=True)
      (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
    (classifier): Sequential(
      (0): Linear(in_features=25088, out_features=4096, bias=True)
      (1): ReLU(inplace=True)
      (2): Dropout(p=0.5, inplace=False)
      (3): Linear(in_features=4096, out_features=4096, bias=True)
      (4): ReLU(inplace=True)
      (5): Dropout(p=0.5, inplace=False)
      (6): Linear(in_features=4096, out_features=1000, bias=True)
    )
  )
  (fc1): Sequential(
    (0): Linear(in_features=1000, out_features=256, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
  )
  (fc2): Linear(in_features=256, out_features=50, bias=True)
)
```

2.

The accuracy of the validation set is 74.16%

```
全部資料 :  2500
我答對了 :  1854
答對率 :  0.7416
```

3.



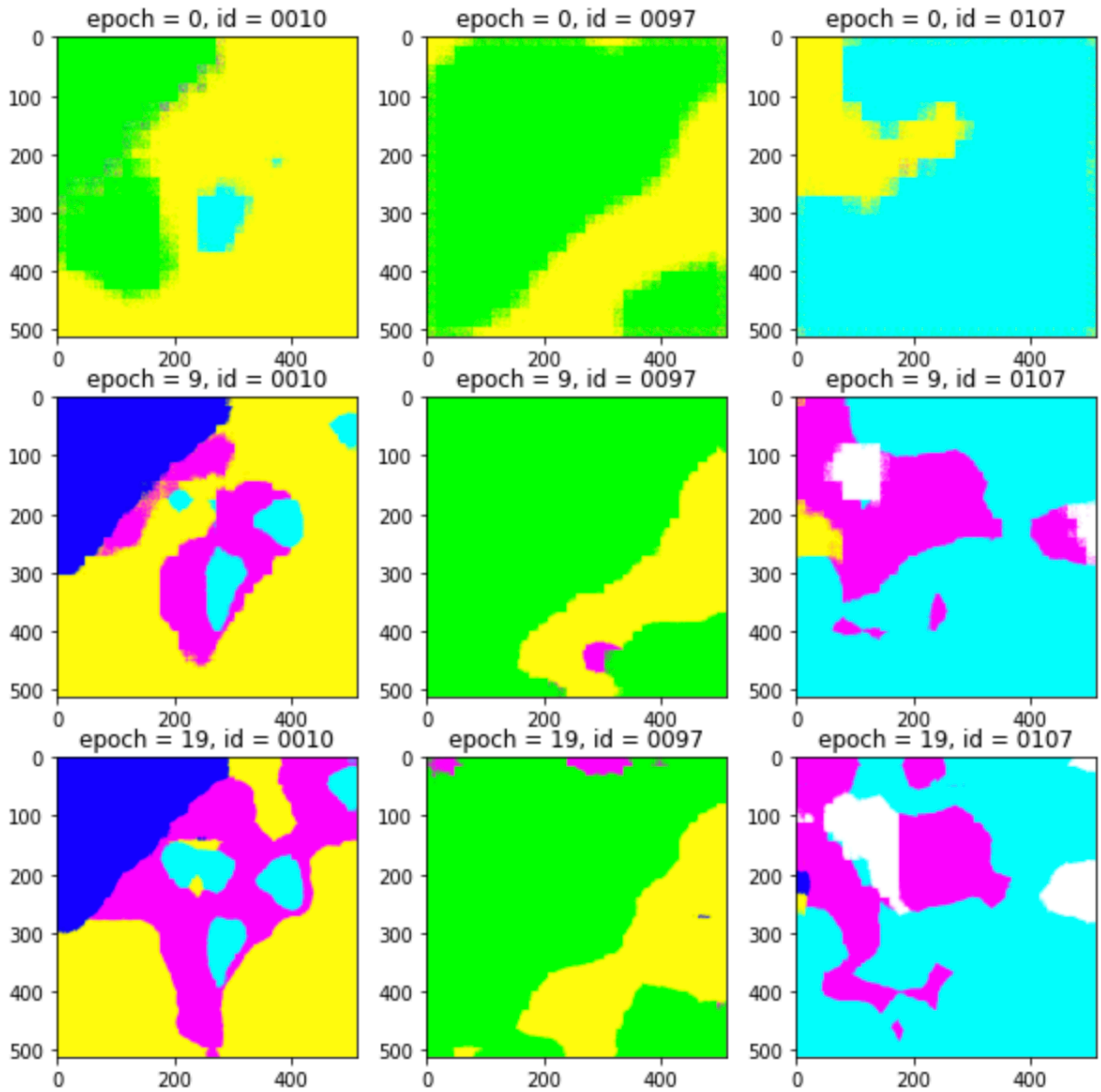
我將model中的layer1的output輸出並concate起來，以validation來說結果會是一個(2500,1000)的矩陣，接著利用TSNE將dim=1000降到dim=2，最後利用plt畫出2500個點的分佈，由上圖可以看出有些點會形成一個聚落，因為此題的class=50，因此其實在上圖中會有超級多個小聚落，每個聚落就是我的model feature越相近也就是同一個class

Problem2 (no collaboration)

1.

```
Net(
  (layer1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fcn): Sequential(
    (0): Conv2d(512, 4096, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
    (7): ReLU()
    (8): ConvTranspose2d(7, 7, kernel_size=(64, 64), stride=(32, 32), padding=(16, 16))
  )
)
```

2.



epoch = 0, id = 0010
epoch = 0, id = 0097
epoch = 0, id = 0107
epoch = 9, id = 0010
epoch = 9, id = 0097
epoch = 9, id = 0107
epoch = 19, id = 0010
epoch = 19, id = 0097
epoch = 19, id = 0107
0
100
200
300
400
500
0
200
400

3.

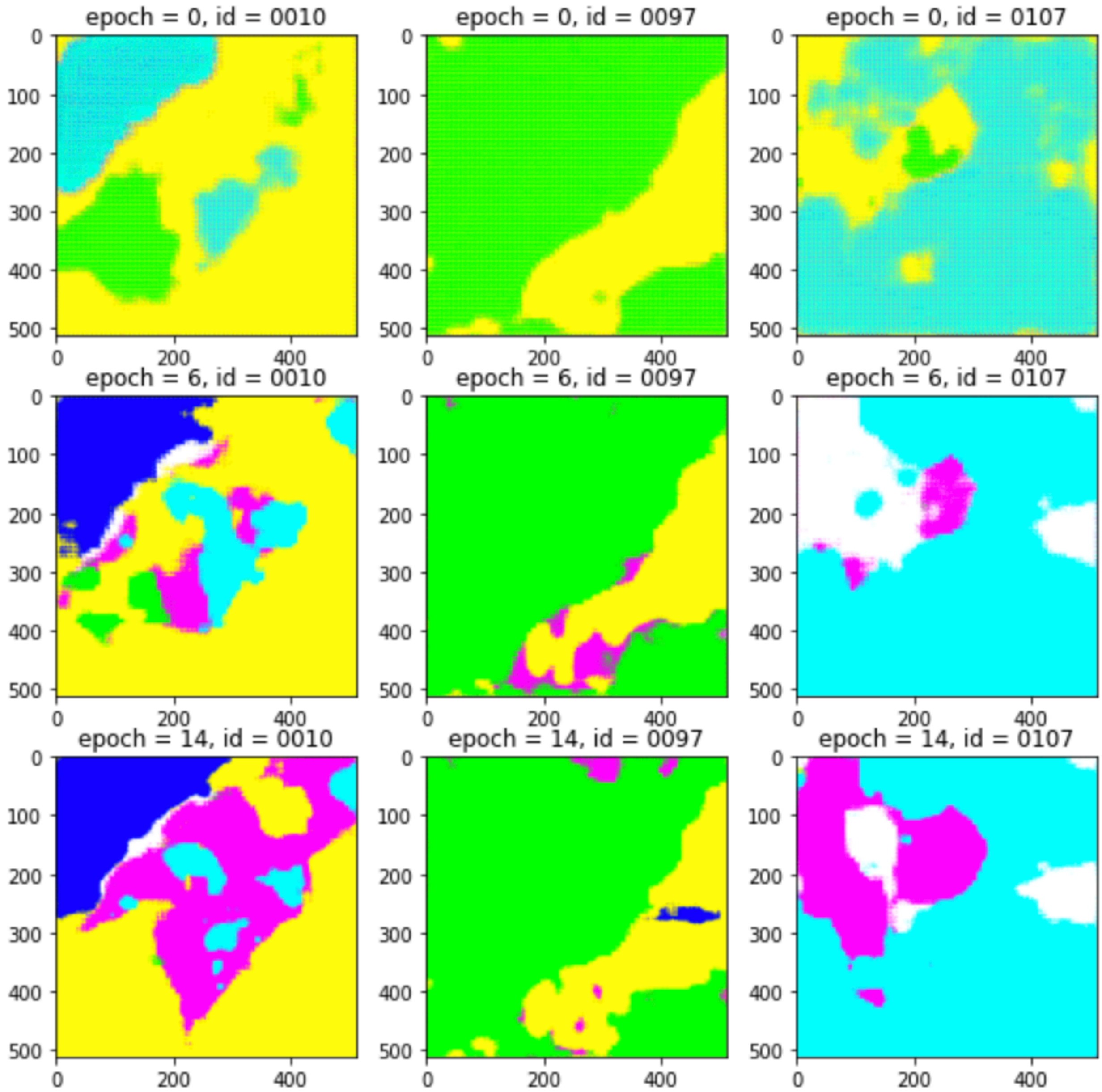
以下附上code中的forward以及print出來的model，方便助教理解

```python
class Net(nn.Module):
    def __init__(self, model):
        super(Net, self).__init__()
        self.layer1 = nn.Sequential(*list(model.children())[:-2][0][:-14])
        self.layer2 = nn.Sequential(*list(model.children())[:-2][0][17:24])
        self.layer3 = nn.Sequential(*list(model.children())[:-2][0][24:])
        self.x2 = nn.Sequential(
            nn.ConvTranspose2d(512,256,4,2,1),
            nn.ReLU(inplace=True)
        )
        self.x4 = nn.Sequential(
            nn.ConvTranspose2d(512,256,8,4,2),
            nn.ReLU(inplace=True)
        )
        self.fcn8 = nn.Sequential(
            nn.Conv2d(256, 4096, 3, padding=1),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Conv2d(4096,4096,1),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Conv2d(4096,7,1),
            nn.ReLU(),
            nn.ConvTranspose2d(7,7,16,8,4)
        )

    def forward(self, x):
        x = self.layer1(x)
        y = x
        x = self.layer2(x)
        z = x
        x = self.layer3(x)
        x = self.x4(x) + self.x2(z) + y
        x = self.fcn8(x)
        return x
```

3.

```
Net(
  (layer1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (x2): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): ReLU(inplace=True)
  )
  (x4): Sequential(
    (0): ConvTranspose2d(512, 256, kernel_size=(8, 8), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
  )
  (fcn8): Sequential(
    (0): Conv2d(256, 4096, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Conv2d(4096, 4096, kernel_size=(1, 1), stride=(1, 1))
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Conv2d(4096, 7, kernel_size=(1, 1), stride=(1, 1))
    (7): ReLU()
    (8): ConvTranspose2d(7, 7, kernel_size=(16, 16), stride=(8, 8), padding=(4, 4))
  )
)
```

4.



5. **baseline model** **improve model**

```
class #0 : 0.75042
class #1 : 0.87743
class #2 : 0.29640
class #3 : 0.80373
class #4 : 0.70067
class #5 : 0.65458

mean_iou: 0.680536
```

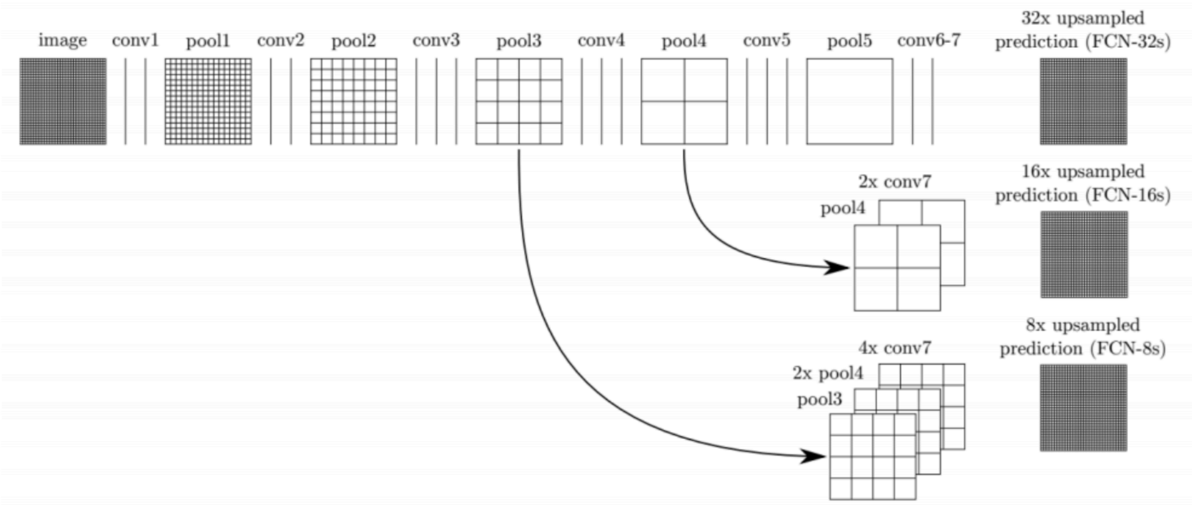
```
class #0 : 0.74603
class #1 : 0.87214
class #2 : 0.34320
class #3 : 0.79568
class #4 : 0.76474
class #5 : 0.67040

mean_iou: 0.698698
```

上圖分別是兩個model的mean_iou：

baseline = 68.05%

improve = 69.87%

我在improve中是使用fcn8對比baseline中使用的fcn32，improve可以觀察更細節的東西，因為是三個階段的feature相加起來的，架構如同下方的FCN-8s:

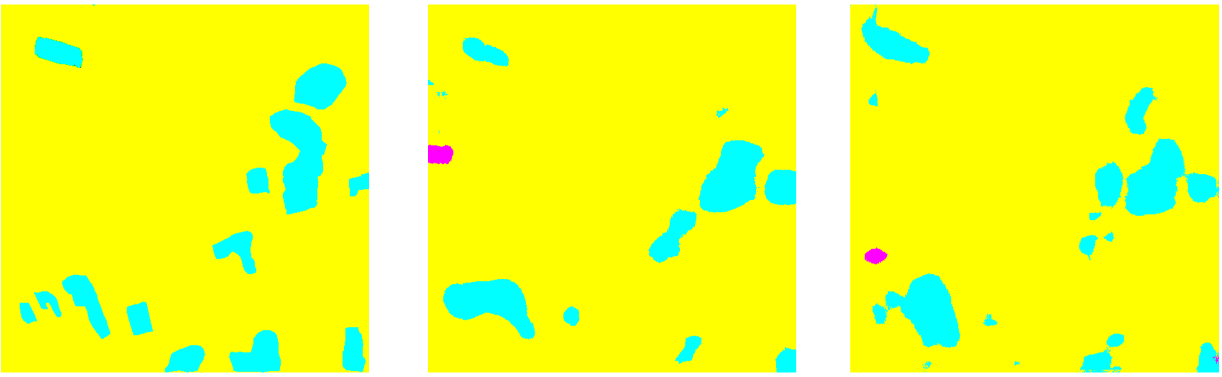


從2.4.中可以發現，improve model確實可以觀察更細節的東西，下圖為我把兩個model最終輸出的mask視覺化後比對真正的label，來驗證improve model確實能關注更細微的東西：

由左至右的順序為 : label —-> baseline —-> improve

id:0010

id :0067



id :0145

id :0244